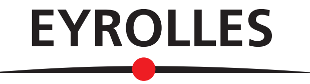
VBA pour Access 2007 & 2010

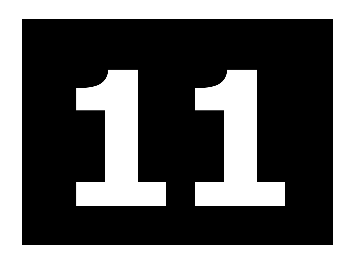
Guide de formation avec cas pratiques

Daniel-Jean David





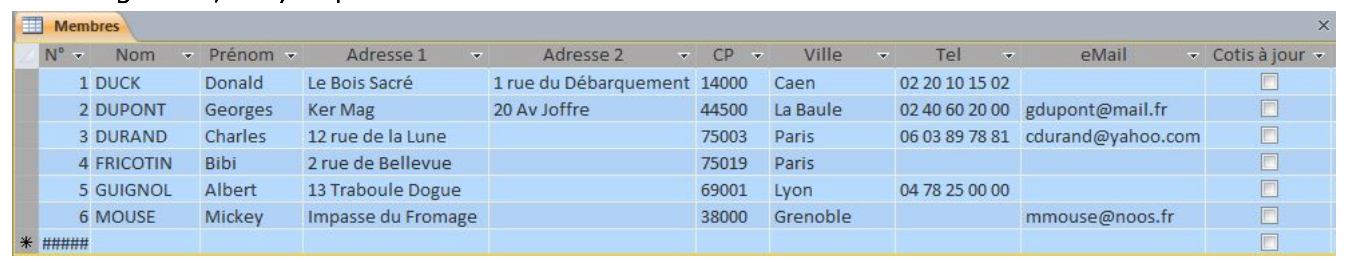
Gestion d'une association



1. LE PROBLÈME

Nous allons gérer l'association des Amis des Animaux, c'est-à-dire inscrire les nouveaux membres, modifier leurs données, en supprimer, etc. Comme utilisation, nous allons créer la page WEB qui affichera le tableau des membres. D'autres utilisations sont envisageables, comme comptabiliser les cotisations, etc., nous les laissons de côté.

La base de données *AmisAnimaux.accdb* contiendra une seule table *Membres*. Dans la première étape, nous produisons le fichier .htm à partir de la BD telle qu'elle est. Les étapes suivantes feront évoluer la base. Dans l'état de départ *AmisAnimaux_00.accdb* du fichier que vous avez en téléchargement, il n'y a que la table Membres avec les données suivantes :



Membres

N°	Nom	Prénom	Adresse 1	Adresse 2	CP	Ville	Tel	eMail	Cotis à jour
1	DUCK	Donald	Le Bois Sacré	1 rue du Débarquement	14000	Caen	02 20 10 15 02		☐
2	DUPONT	Georges	Ker Mag	20 Av Joffre	44500	La Baule	02 40 60 20 00	gdupont@mail.fr	☐
3	DURAND	Charles	12 rue de la Lune		75003	Paris	06 03 89 78 81	cdurand@yahoo.com	☐
4	FRICOTIN	Bibi	2 rue de Bellevue		75019	Paris			☐
5	GUIGNOL	Albert	13 Traboule Dogue		69001	Lyon	04 78 25 00 00		☐
6	MOUSE	Mickey	Impasse du Fromage		38000	Grenoble		mmouse@noos.fr	☐
#####									☐

La rubrique <Cotis. à jour> est prévue, mais elle ne sera pas gérée ici. A part N° qui est en entier à incrémentation automatique, tous les champs sont en texte.

Pour obtenir le fichier *AmisAnimaux_0.accdb*, vous créez le formulaire nommé *Menu*, qui propose un bouton par fonctionnalité comme suggéré au chapitre 10 :



On voit sur la figure qu'après l'explication succincte à côté de chaque bouton, figure le nom de la procédure associée dans Module1 ; pour le moment ces procédures sont toutes vides. Voici le module de classe du formulaire :

```
Private Sub B_HTM_Click()
  GenerHTM
End Sub

Private Sub B_ModifSupp_Click()
  ModifMembre
End Sub

Private Sub B_NouvMemb_Click()
  NouvMembre
End Sub

Private Sub B_Quit_Click()
  DoCmd.Close acForm, "Menu", acSaveNo
  DoCmd.Quit acQuitSaveAll
End Sub
```

Utilisation des fichiers téléchargés

Chaque cas pratique a son dossier. Les fichiers des cas pratiques ont leur nom terminé par le n° d'étape. Pour reproduire le passage de l'étape n à n+1, vous copiez le fichier n dans un nouveau répertoire, ainsi que les fichiers auxiliaires éventuels. Ensuite, copiez le fichier n sous le nom n+1 et faites les manipulations sur ce fichier n+1.

2. ROUTINE D'INITIALISATION

Nous commençons par l'introduction d'une routine d'initialisation `InitGen` et de quelques variables: `Chem` (le dossier des fichiers : au départ, le fichier .htm sera dans le même que la base de données), `Sep` (le séparateur \ ou : - il faut changer l'instruction Sep = … sur Mac et c'est probablement la seule à changer), `NbRub` (nombre de rubriques traitées : on le diminue de 1 car on ne s'occupe pas de la cotisation ➍) et le tableau `NomsRub` (les noms de rubriques). On introduit une variable `Cnx` pour la connexion, `Rst` pour le `Recordset` de la table des membres et `NbLig` le nombre d'enregistrements. Celui-ci est déterminé par `RecordCount` ➋ car le `Recordset` est ouvert en lecture seule ➊.

Les routines qui suivent doivent être saisies dans le module *Module 1*.

Voici la routine `InitGen` précédée des déclarations.

```
'--------Version 1.0------------
Public Chem As String, Sep As String, Rub As String
Public Cnx As ADODB.Connection, Rst As ADODB.Recordset
Public NbRub As Integer, NomsRub(10) As String, FF As Object
Public NbLig As Integer, Kol As Integer, Lig As Integer

Sub InitGen()
  Sep = "\" ' Remplacer par Sep=":" sur Mac
  Chem = CurrentProject.Path + Sep
  Set Cnx = CurrentProject.Connection
  Set Rst = New ADODB.Recordset
  Rst.Open "Membres", Cnx, adOpenStatic, adLockReadOnly ➊
  NbLig = Rst.RecordCount ➋
  NbRub = 0
  For Each FF In Rst.Fields ➌
    NbRub = NbRub + 1
    NomsRub(NbRub) = FF.Name
  Next
  NbRub = NbRub - 1 ➍
End Sub
```

En ❸, on détermine les noms de rubrique par balayage de la collection `Fields`.

3. CONSTRUCTION DU FICHIER .HTM

La structure est très simple : début de la page Web, tableau des membres, fin de la page. Le tableau a lui-même un début et une fin entourant une double structure répétitive pour les lignes (les membres) et les colonnes (les rubriques). Voici le texte HTML représentant le tableau à afficher :

```
<html><head>                                |début de la page
<title>Les Amis des Animaux</title>                    |
</head><body><center>                       |
<h2>Membres de l'Association</h2>                |
<h2>Les Amis des Animaux</h2></center>           |
<table border width=95%>                         |     |début du tableau
<tr><td>nom de rubrique …</td></tr>              |     |     |ligne d'en-tête
<tr>                                        |    |     |ligne
<td>rubrique</td>                           |    |     |     |rubrique
<td>rubrique</td>                           |    |     |     |rubrique
</tr>                                       |    |     |fin de la ligne
</table>                                         |     |fin du tableau
</body></html>                              |fin de la page
```

Voici la page Web correspondant au fichier *Membres.htm* obtenu à partir des données page 170 grâce à la procédure `GenerHTM` ci-après.

Membres de l'Association

Les Amis des Animaux

N°	Nom	Prénom	Adresse 1	Adresse 2	CP	Ville	Tel	eMail
1	DUCK	Donald	Le Bois Sacré	1 rue du Débarquement	14000	Caen	02 20 10 15 02	
2	DUPONT	Georges	Ker Mag	20 Av Joffre	44500	La Baule	02 40 60 20 00	eMail
3	DURAND	Charles	12 rue de la Lune		75003	Paris	06 03 89 78 81	eMail
4	FRICOTIN	Bibi	2 rue de Bellevue		75019	Paris		
5	GUIGNOL	Albert	13 Traboule Dogue		69001	Lyon	04 78 25 00 00	
7	MOUSE	Mickey	Impasse du Fromage		38000	Grenoble		eMail

```
'-----------------------------------------------------GenerHTM--
Sub GenerHTM()
  InitGen
  Open Chem + "Membres.htm" For Output As #1
  Print #1, "<html><head>" + vbCr;  ' Début page
  Print #1, "<title>Les Amis des Animaux</title>" + vbCr;
  Print #1, "</head><body><center>" + vbCr;
  Print #1, "<h2>Membres de l'Association</h2>" + vbCr;
  Print #1, "<h2>Les Amis des Animaux</h2></center>" + vbCr;
  Print #1, "<table border width=95%>" + vbCr; ' Début tableau
  Print #1, "<tr>" + vbCr;
  For Kol = 1 To NbRub                   ' Noms de rubriques
    Print #1, "<td><b>" + NomsRub(Kol) + "</b>" + "</td>" + vbCr;
  Next Kol
  Print #1, "</tr>" + vbCr;
```

```
  Rst.MoveFirst
  For Lig = 1 To NbLig
    Print #1, "<tr>" + vbCr;
    For Kol = 1 To NbRub - 1          'Les rubriques
      Rub = CStr(Nz(Rst.Fields(NomsRub(Kol)).Value, ""))
      If Rub = "" Then Rub = " "
      Print #1, "<td>" + Rub + "</td>" + vbCr;
    Next Kol
    Rub = CStr(Nz(Rst.Fields(NomsRub(NbRub)).Value, "")) 'eMail à part
    If Rub = "" Then Rub = " " Else _
         Rub = "<a href=""mailto:" + Rub + """>eMail</a>"
    Print #1, "<td>" + Rub + "</td>" + vbCr;
    Print #1, "</tr>" + vbCr;
    Rst.MoveNext
  Next Lig
  Print #1, "</table>" + vbCr;
  Print #1, "</body></html>" + vbCr;
  Close 1
  Rst.Close: Cnx.Close
  Set Rst = Nothing: Set Cnx = Nothing
End Sub
```

La procédure GenerHTM commence par appeler InitGen. Ensuite, chaque ligne d'écriture html se fait par un print # de la chaîne de caractères voulue ; on termine par vbCr et **;** pour avoir un parfait contrôle des lignes. Si la rubrique est vide, on met " " (l'espace en HTML) pour assurer la continuité de la bordure. Pour la rubrique eMail, la chaîne est :

`"<a href=""mailto:adresse"">eMail</a>"` : remarquez les doubles guillemets pour incorporer un guillemet.

On utilise les variables `Lig` (numéro de ligne), `Kol` (numéro de rubrique/colonne) et `Rub` (le texte de la rubrique). Celui-ci est obtenu dans l'enregistrement courant du Recordset à partir du champ correspondant au nom de rubrique.

Le `<b> </b>` entourant l'écriture de chaque nom de rubrique, le met en gras.

Le nom du fichier Web produit est fixé à *Membres.htm* dans l'instruction `Open`. Une telle chose est en principe à éviter : on doit paramétrer au maximum. Dans notre exemple, on pourrait introduire une variable `NomFichWeb` obtenue par une InputBox :

```
NomFichWeb=InputBox("Nom du fichier Web à créer ? ",,"Membres.htm")
```

Nous vous laissons l'implantation complète à titre d'exercice complémentaire.

Notez aussi dans ce contexte qu'on ne se préoccupe pas de savoir s'il existe déjà un fichier .htm de même nom dans le dossier : si c'est le cas, ce fichier est écrasé et remplacé par le fichier que vous créez, sans qu'il y ait le moindre message pour vous avertir. Dans ce cas, le fichier .htm reflètera l'état actuel de la table Membres au moment de l'exécution de la commande et c'est probablement ce qui est souhaité. Si vous voulez conserver un état antérieur de la page Web, il faut préalablement sauvegarder le fichier *Membres.htm*.

Après l'ouverture du fichier, une batterie de `Print #` écrit les lignes de début du fichier .htm tel qu'esquissé page 200. On implante la balise de début de tableau et la boucle `For Kol`…. remplit la 1re ligne avec les noms de rubrique.

La boucle `For Lig =` … parcourt les enregistrements du `Recordset` (Méthodes `MoveFirst` au départ, puis `MoveNext` dans la boucle). Pour chaque enregistrement d'un membre de l'association, il y aura une ligne du tableau, donc on implante la balise `<tr>` de début de ligne. On a ensuite la boucle sur les rubriques. On termine par la balise `</tr>` de fin de ligne.

Les rubriques sont traitées en deux temps : les `NbRub-1` premières rubriques sont traitées dans la boucle `For Col…` Le contenu trouvé sur la feuille est converti en chaîne. S'il est vide, on inscrira ` ` qui figure un espace : si on ne le faisait pas, on aurait une case vide dans le tableau et la bordure aurait une discontinuité inesthétique (ceci est un problème HTML qui sort du sujet de ce livre). Bien sûr, chaque rubrique est annoncée par la balise <td>. Remarquez l'emploi de `Nz` car une rubrique non renseignée dans la table donne la valeur `Null`.

La dernière rubrique est traitée à part car il n'y a pas qu'à recopier l'adresse eMail, il faut construire le lien en insérant le contenu lu sur la feuille des membres entre les balises `<a>` et `</a>`.

Le programme se termine par les balises de fin de tableau et de fin d'HTML et, surtout, par la fermeture du fichier à ne pas oublier sous peine d'écriture incomplète.

Une fois la frappe finie, faites *Débogage – Compiler..*. Un certain nombre d'erreurs peuvent vous être signalées : comparez avec le listing ci-dessus et corrigez. Sauvegardez la base de données (sous le nom *AmisAnimaux_1.accdb*).

Attention, vous ne devez pas écraser le fichier de même nom téléchargé. Vous devez avoir conservé une copie des fichiers originaux téléchargés dans un autre dossier.

Il vous reste à tester l'exécution, ce qui s'obtient en cliquant sur le bouton Génère HTM de la feuille *Menu*. Si vous n'avez pas fait d'erreur, vous devriez obtenir un fichier *Membres.htm* et la visualisation par votre navigateur doit avoir l'aspect de la figure de la page 172.

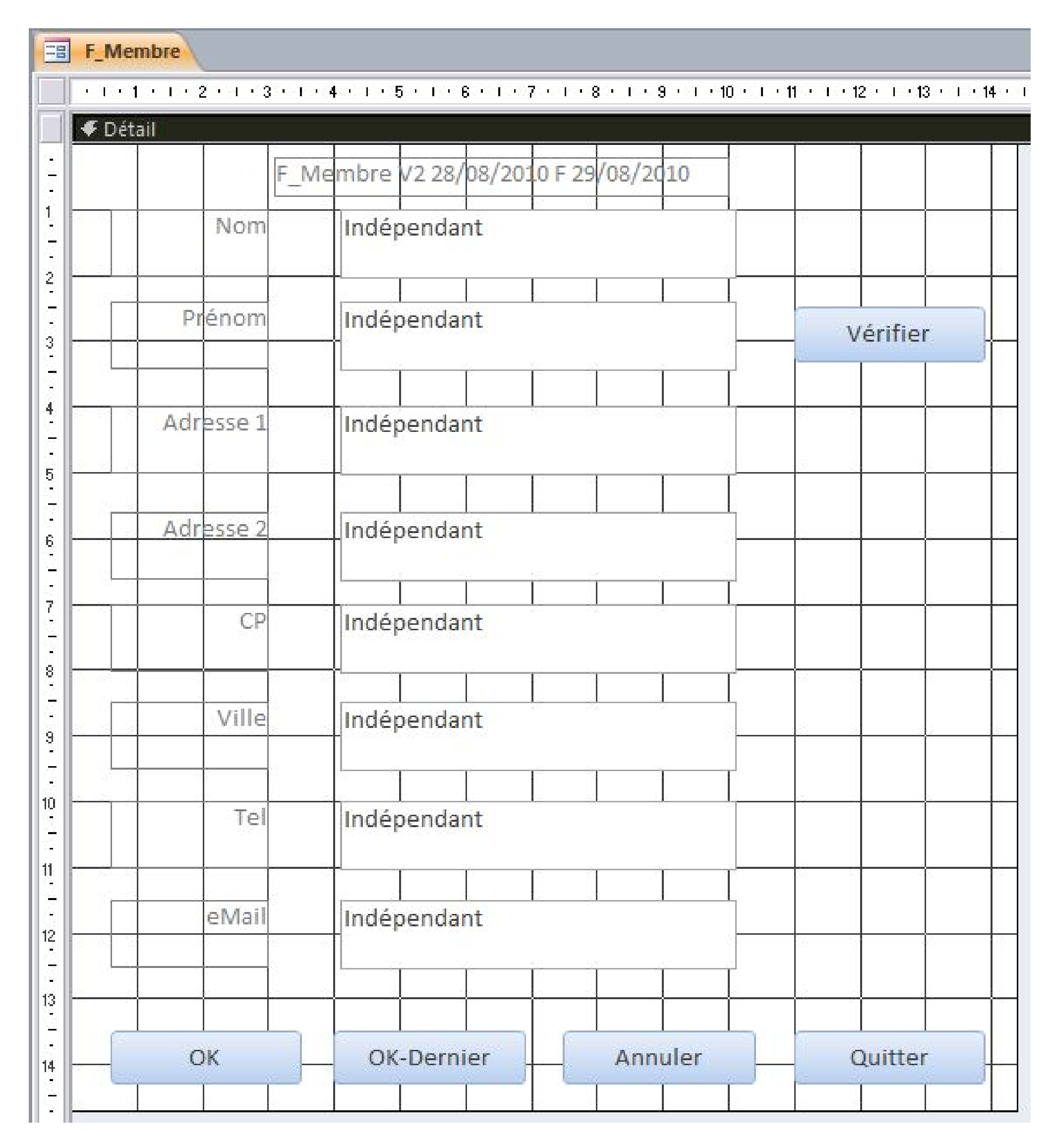
Remarquez que nous précédons chaque procédure d'un commentaire formé de tirets qui servira de séparateur si vous imprimez le listing. En effet, à l'impression, les traits de séparation installés par l'Editeur VBA n'apparaissent pas. Le rappel du nom de la procédure à la fin aide à la repérer si le listing est très long.

---

1. CRÉER UN FORMULAIRE

Nous passons à la gestion de la base de données, et, d'abord, à l'entrée d'un nouveau membre. Il nous faut donc un formulaire pour entrer ses données.

- Copiez la base *AmisAnimaux_1. accdb* en *AmisAnimaux_2.accdb* sur laquelle vous allez travailler.
- Passez en mode création de formulaire ; nommez-le F_Membre, Caption Nouveau Membre. Fixez les propriétés Fen Indépendante, Fen Modale à Oui, Boutons de déplacement à Non.
- Créez une TextBox avec son Label à côté ; donnez la valeur « Droite » à la propriété Aligner le texte du label ; sélectionnez les deux et faites Copier.
- Faites Coller 7 fois : vous avez 8 couples (on gère 8 rubriques).
- Donnez aux labels les `Captions` respectives *Nom*, *Prénom* … (les noms de rubriques) ; donnez aux TextBox les noms TB_Nom *etc.* (les noms des rubriques).
- Créez 5 boutons, les 4 premiers en bas, le 5[e] à côté du prénom. Donnez les Name (et Caption) respectifs B_OK (OK), B_OKDern (OK-Dernier), B_Annul (Annuler), B_Quit (Quitter) et B-Ver (Vérifier).
- Créez un Label en haut avec `Visible` = `False` et la `Caption = F_Membre V2 date F date` : ce label apparaîtra au listing alors que le titre du formulaire n'apparaît pas. Le formulaire doit avoir l'aspect :



Le bouton Vérifier devra être cliqué après avoir entré nom et prénom : le système préviendra si nom et prénom identiques se trouvent déjà dans la base. Les boutons de validation ne seront activés qu'après cette vérification. La dualité OK, Annuler /OK Dernier, Quitter permet d'entrer une série de membres : pour le dernier, on valide par OK-Dernier.

Cette gestion utilise deux booléens `Satisf` et `Dernier` : `Satisf` est vrai si on a validé les données d'un membre, `Dernier` si c'est le dernier de la série. On a en plus une variable `Mode` qui distinguera le cas Nouveau membre du cas Modification, car, par économie, nous utiliserons le même formulaire, à peine modifié. Ces variables sont publiques, ainsi que le tableau `DonMemb` des données du membre qui sera géré comme objet `Scripting.Dictionary`.

En résumé, il s'ajoute en tête du Module 1 les déclarations :

```
Public Mode As Integer, Satisf As Boolean, Dernier As Boolean, DonMemb As Object
```

2. MODULE DE CLASSE DU FORMULAIRE

Ce module est essentiellement formé des procédures événements des contrôles du formulaire, mais il peut s'ajouter d'autres procédures si, comme ce sera le cas ici, une même opération est à effectuer à partir de plusieurs contrôles.

Pour implanter une procédure événement, vous sélectionnez le contrôle puis cliquez sur la ligne Sur événement dans l'onglet Événements la fenêtre de propriétés ; choisissez Générateur de code. Il s'implante souvent inopinément des routines `_Click`, laissées vides : pensez à les supprimer.

Le module a deux variables globales au niveau module Nm et Pr qui contiendront le nom et le prénom du membre en cours d'ajout.

Nous avons d'abord la routine `Form_Current` où nous n'implantons que la branche Mode=0 : on ne fait qu'initialiser Nm et Pr et fixer le titre du formulaire et la légende du bouton « Vérifier ».

Lorsqu'on a entré un nom et/ou un prénom, on désactive les boutons « OK », car on doit effectuer la vérification, d'où les deux routines `TB_Nom_Exit` et `TB_Prénom_Exit`. Elles prennent en compte respectivement le nom et le prénom entrés.

Les quatre routines des boutons de validation `B_Annul_Click`, `B_OK_Click`, `B_OKDern_Click` et `B_Quit_Click` sont très semblables : avant de fermer le formulaire elles fixent en conséquence les booléens sur lesquels est basée la gestion : `Dernier` est mis à vrai pour les boutons qui terminent une série B_OKDern et B_Quitter. `Satisf` est mis à faux par les boutons d'annulation et à vrai par les boutons OK.

Les boutons « OK » appellent la procédure `CaptureDon` qui transfère les données des contrôles dans le tableau `DonMemb`. En effet, si on clique sur « OK », c'est que les données entrées dans les contrôles sont correctes. Le tableau Dictionnaire `DonMemb` sert à les mémoriser pour récupération dans Module 1 ; il est géré comme vu au chapitre 10 dans la section sur le dictionnaire de données. Mais avant, d'appeler `CaptureDon`, les routines font appel à la fonction booléenne `VerNomPren` qui prend la valeur `False` s'il manque soit le nom soit le prénom : dans ce cas, inutile de capturer les données et on ne ferme pas le formulaire.

La routine `B_Ver_Click` est la plus délicate. Pour le moment, nous n'implantons que la branche Mode=0. Pour vérifier s'il y a déjà un membre ayant à la fois même nom et même prénom, on crée un objet commande qui exécutera une requête de comptage des enregistrements satisfaisant à cette condition. Le résultat de la requête est Res(0). Value.

Si ce résultat est différent de 0, on demande à l'utilisateur s'il veut tout de même entrer ce membre (deux membres peuvent avoir mêmes nom et prénom ; espérons qu'ils n'ont pas la même adresse !) et alors on active aussi les boutons OK. Si la réponse est non, l'utilisateur doit changer le nom et/ou le prénom ou bien annuler.

```
Dim Nm As String, Pr As String

Sub CaptureDon()
  Dim NomTB As String, ct As Control, Tx As String
  For Each ct In Controls
    If Left(ct.Name, 3) = "TB_" Then
      NomTB = Mid(ct.Name, 4)
      ct.SetFocus
      Tx = Nz(ct.Text, "")
      DonMemb.Add NomTB, Tx
    End If
  Next
End Sub

Function VerNomPren() As Boolean
  If (Nm = "") Or (Pr = "") Then
    MsgBox "Il faut un nom et un prénom"
    VerNomPren = False
  Else
    VerNomPren = True
  End If
End Function

Private Sub B_Annul_Click()
  Dernier = False
  Satisf = False
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub B_OK_Click()
  If Not VerNomPren Then Exit Sub
  CaptureDon
  Dernier = False
  Satisf = True
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub B_OKDern_Click()
  If Not VerNomPren Then Exit Sub
  CaptureDon
  Dernier = True
  Satisf = True
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub B_Quit_Click()
  Dernier = True
  Satisf = False
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub
```

```
Private Sub B_Ver_Click()
  Dim Cm As New ADODB.Command, Res, Rep
  If Mode = 0 Then
    Set Cm.ActiveConnection = CurrentProject.Connection
    Cm.CommandType = adCmdText
    Cm.CommandText = "select count(Nom) from Membres " + _
        "where (Nom='" + Nm + "') and (Prénom='" + Pr + "')"
    Set Res = Cm.Execute
    If Res(0).Value <> 0 Then
      Rep = MsgBox("Ce nom et prénom sont déjà présents" + vbCr + _
        "voulez-vous tout de même entrer ce membre", vbYesNo + _
        vbExclamation)
      If Rep = vbYes Then B_OK.Enabled = True: B_OKDern.Enabled = True
    Else
      B_OK.Enabled = True: B_OKDern.Enabled = True
    End If
  Else
  ' laissé vide pour le moment
  End If
End Sub

Private Sub Form_Current()
  Nm = "": Pr = ""
  If Mode = 0 Then
    Me.Caption = "Nouveau membre"
    B_Ver.Caption = "Vérifier"
  Else
  ' laissé vide pour le moment
  End If
End Sub

Private Sub TB_Nom_Exit(Cancel As Integer)
  Nm = Nz(TB_Nom.Text, "")
  B_OK.Enabled = False
  B_OKDern.Enabled = False
End Sub

Private Sub TB_Prénom_Exit(Cancel As Integer)
  Pr = Nz(TB_Prénom.Text, "")
  B_OK.Enabled = False
  B_OKDern.Enabled = False
End Sub
```

3. PROCÉDURE NOUVMEMBRE DANS MODULE 1

```
'-----------------------------------------------------NouvMembre--
Sub NouvMembre()
  Dim NomC As String, ff As Object
  Dernier = False
  While Not Dernier
    Set DonMemb = CreateObject("Scripting.Dictionary")
    Set Cnx = CurrentProject.Connection
    Set Rst = New ADODB.Recordset
    Rst.Open "Membres", Cnx, adOpenDynamic, adLockOptimistic
    Satisf = False
```

```
     Mode = 0
     DoCmd.OpenForm "F_Membre", , , , acFormAdd, acDialog
     If Satisf Then
       Rst.AddNew
       For Each ff In Rst.Fields
         NomC = ff.Name
         If (NomC <> "N°") And (NomC <> "Cotis à jour") _
             Then ff.Value = DonMemb(NomC)
       Next
       Rst.Update
     End If
     DonMemb.RemoveAll: Set DonMemb = Nothing
     Rst.Close: Cnx.Close
     Set Rst = Nothing: Set Cnx = Nothing
   Wend
End Sub
```

La structure de `NouvMemb` est en fait simple :

```
Dernier=False
While Not Dernier        'Tant qu'on n'a pas entré le dernier de la série
|   Set DonMemb..    'Préparer le dictionnaire des données
|   Rst.Open…        'Ouvrir le Recordset
|      DoCmd.OpenForm     'Afficher le formulaire
|       If Satisf Then            'Si on a obtenu une donnée correcte
|       |     Rst.AddNew   'Crée nouvel enregistrement
|    |     For Each ff     'Remplissage des champs
|       |     |…
|       |     Next
|       |     Rst.Update   'Installe le nouvel enregistrement
|       End If
|    'Fait le ménage en vue d'un éventuel enregistrement suivant
Wend
```

La boucle `For Each ff` suit exactement le schéma de remplissage des champs vu au chapitre 10 puisque les TextBox ont été nommées suivant le modèle TB_<nom du champ>.

Sauvegardez le fichier sous le nom *AmisAnimaux_2.accdb* avec toujours la même précaution d'avoir conservé une copie intacte des fichiers originaux téléchargés.

Pour tester le programme à l'étape 2, vous cliquez sur le bouton « Nouveau membre », vous entrez une série de nouveaux membres (clic sur OK après chaque et sur OK-Dernier après le dernier) : vous devez vérifier que les données des membres sont bien entrées.

4. QUELQUES AMÉLIORATIONS

Nous proposons deux légers changements dans le code qui vont servir en vue de la 3e étape. Cela va constituer le fichier *AmisAnimaux_2a.accdb*. Dans `NouvMembre`, la boucle de remplissage des champs `For Each ff` va servir plusieurs fois. Donc nous l'implantons dans une procédure `RemplitChamps` et la partie concernée du module est maintenant :

```
'----------------------------------------------------RemplitChamps--
Sub RemplitChamps()
  Dim NomC As String, ff As Object
  For Each ff In Rst.Fields
    NomC = ff.Name
    If (NomC <> "N°") And (NomC <> "Cotis à jour") _
            Then ff.Value = DonMemb(NomC)
    Next
End Sub

'----------------------------------------------------NouvMembre--
Sub NouvMembre()
  Dernier = False
  While Not Dernier
    Set DonMemb = CreateObject("Scripting.Dictionary")
    Set Cnx = CurrentProject.Connection
    Set Rst = New ADODB.Recordset
    Rst.Open "Membres", Cnx, adOpenDynamic, adLockOptimistic
    Satisf = False
    Mode = 0
    DoCmd.OpenForm "F_Membre", , , , acFormAdd, acDialog
    If Satisf Then
      Rst.AddNew
      RemplitChamps
      Rst.Update
    End If
    DonMemb.RemoveAll: Set DonMemb = Nothing
    Rst.Close: Cnx.Close
    Set Rst = Nothing: Set Cnx = Nothing
  Wend
End Sub
```

De même, dans `B_Ver_Click` du module associé au formulaire, la partie de test de la présence du nom et prénom est susceptible de généralisation. Nous créons une fonction booléenne `TestSQL` qui a le texte d'une requête comme argument et qui renvoie vrai s'il existe des enregistrements satisfaisant à la requête.

```
Function TestSQL(chSQL As String) As Boolean
  Dim Cm As New ADODB.Command, Res
  Set Cm.ActiveConnection = CurrentProject.Connection
  Cm.CommandType = adCmdText
  Cm.CommandText = chSQL
  Set Res = Cm.Execute
  If Res(0).Value <> 0 Then TestSQL = True Else TestSQL = False
  Set Cm = Nothing
End Function

Private Sub B_Ver_Click()
  Dim Ch As String, Rep
  If Mode = 0 Then
    Ch = "select count(Nom) from Membres " + _
        "where (Nom='" + Nm + "') and (Prénom='" + Pr + "')"
```

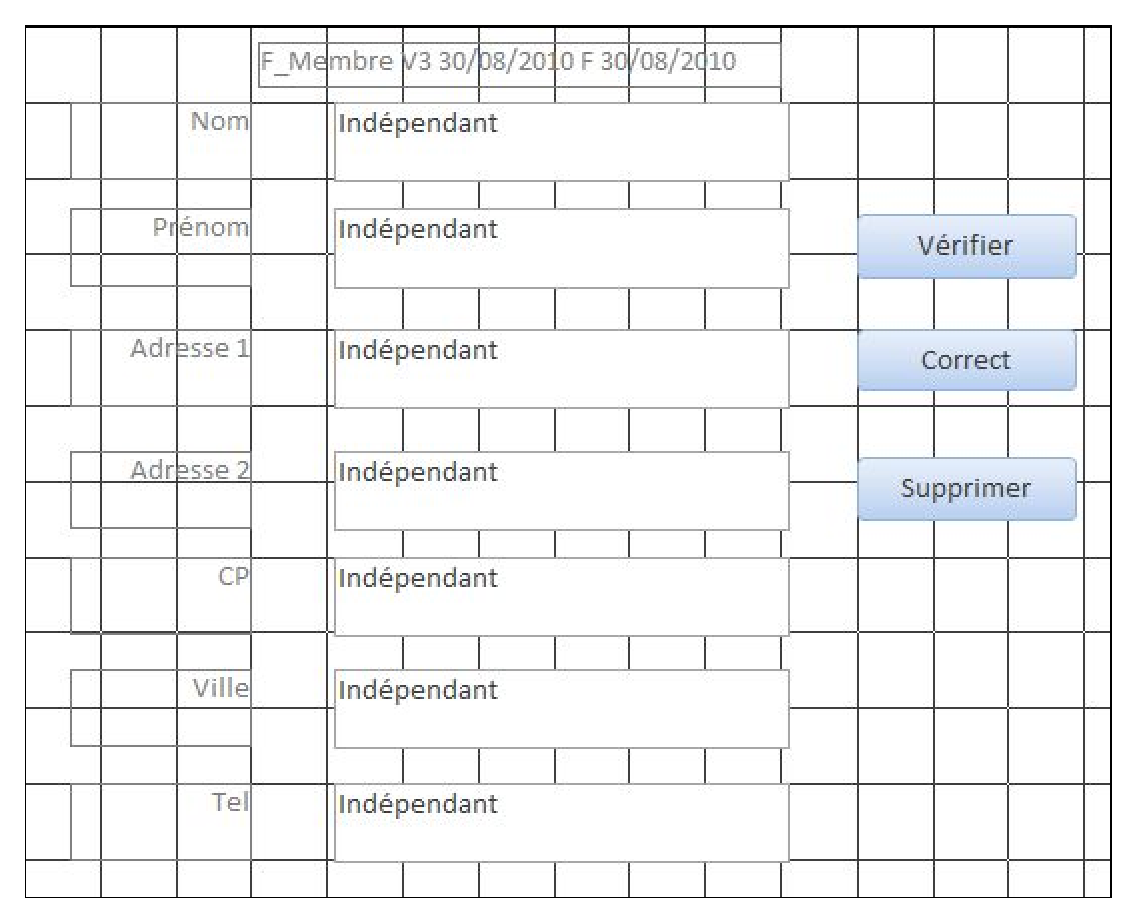
```
    If TestSQL(Ch) Then
      Rep = MsgBox("Ce nom et prénom sont déjà présents" + vbCr + _
        "voulez-vous tout de même entrer ce membre", vbYesNo + _
        vbExclamation)
      If Rep = vbYes Then B_OK.Enabled = True: B_OKDern.Enabled = True
    Else
      B_OK.Enabled = True: B_OKDern.Enabled = True
    End If
  Else
  ' laissé vide pour le moment
  End If
End Sub
```

ÉTAPE 3 – MODIFICATION/SUPPRESSION

1. CONSTRUIRE LE FORMULAIRE

Copiez *AmisAnimaux_2a.accdb* en *AmisAnimaux_3.accdb* sur lequel vous allez travailler.

Pour la modification, le problème est de trouver l'enregistrement à modifier. On fait la recherche sur le nom : lorsqu'on a trouvé une concordance, on affiche l'ensemble des données de l'enregistrement et l'utilisateur doit cliquer sur Correct si c'est l'enregistrement cherché. Sinon, il doit cliquer sur Chercher/Suivant car il peut y avoir plusieurs membres de même nom. Le libellé « Chercher/Suivant » remplace le libellé « Vérifier » ; les deux boutons supplémentaires sont visibles et actifs seulement si Mode=1. Dans ce cas, on change aussi le titre du formulaire dans la routine `Form_Current`. Voici le nouvel aspect du formulaire :



Nous essayons d'avoir un mode d'emploi assez perfectionné pour la recherche de l'enregistrement à modifier. Si vous fournissez le prénom, la recherche se fait sur nom et prénom exacts. Si vous ne fournissez pas le prénom, on accepte tout nom contenant ce que vous avez tapé dans la zone nom.

2. PROCÉDURES DU MODULE DE CLASSE

Dans la fenêtre de code associée au formulaire, vous avez un certain nombre de routines à modifier, et il s'ajoute les routines de clic des deux boutons supplémentaires.

Le module de classe a une nouvelle variable, `EnCours`, vraie si ce n'est pas la 1re fois que vous tapez sur le bouton : la 1re fois, il signifie Chercher, les fois suivantes, il signifie Suivant..

Les routines des quatre boutons de validation ainsi que les deux routines d'Exit des deux TextBox sont inchangées mais `CaptureDon` commence par un vidage du dictionnaire : en effet, maintenant, pour une ouverture du formulaire, il peut y avoir plusieurs enregistrements à regarder lors de la recherche de l'enregistrement à modifier.

`CaptureDon` est complétée par un certain nombre de routines de transfert entre TextBox et le dictionnaire `DonMemb` ou entre champs de l'enregistrement en cours : `VideBDi`, `RemplitBDi`, `RecupChamps` et `RemplitChamps`. Les deux dernières sont dans Module1.

La procédure `Form_Current` a maintenant aussi la branche pour Mode=1 (sous le `Else`). Même la branche Mode = 0 est à modifier puisqu'il s'ajoute la gestion des activations et visibilité des boutons supplémentaires *B_Correct* et *B_Suppr*. Dans la branche Mode=1 on initialise `EnCours` à `False`.

La routine `B_Correct_Click` active le bouton Supprimer et les deux « OK » puisque le membre sur lequel on veut agir est maintenant trouvé.

`B_Suppr_Click` demande une confirmation et, si oui, effectue la suppression. On agit sur l'enregistrement en cours. Remarquez que, après la suppression de l'enregistrement, on appelle `B_Annul_Click` : en effet, au retour dans le programme appelant, tout doit se passer comme si on avait annulé car la modification de la table Membres a été effectuée.

C'est la routine `B_Ver_Click` qui subit les plus importantes modifications. La branche Mode=0 est inchangée, ce qui prouve la solidité de notre programmation. La branche `Else` est subdivisée en deux selon la valeur de `EnCours`. Si `EnCours` vaut 0, on commence une recherche : la chaîne de requête est construite en fonction de la présence ou non du prénom (si le nom est absent, il y a un message d'erreur). On teste alors s'il y a des enregistrements conformes et à ce moment on ouvre le Recordset sur la requête proprement dite.

Si `EnCours` vaut 1, c'est que l'utilisateur a cliqué sur Chercher/Suivant : il faut donc chercher plus loin à condition qu'on ne dépasse pas la fin du `Recordset` ; dans ce cas, un message en avertit l'utilisateur. Si on a pu lire un enregistrement, les appels à `RecupChamps` puis `RemplitBDi` affichent les données : l'utilisateur peut juger s'il doit cliquer sur Correct ou demander l'aller plus loin.

Si on clique sur Correct, les données pourront être modifiées et les valeurs modifiées validées puisque les boutons de validation auront été activés par `B_Correct_Click`.

```
Dim Nm As String, Pr As String
Dim EnCours As Boolean

Sub CaptureDon()
  Dim NomTB As String, ct As Control, Tx As String
  DonMemb.RemoveAll
  For Each ct In Controls
    If Left(ct.Name, 3) = "TB_" Then
      NomTB = Mid(ct.Name, 4)
      ct.SetFocus
      Tx = Nz(ct.Text, "")
      DonMemb.Add NomTB, Tx
    End If
  Next
End Sub

Sub VideBDi()
  Dim NomTB As String, ct As Control
  DonMemb.RemoveAll
  For Each ct In Controls
    If Left(ct.Name, 3) = "TB_" Then
      NomTB = Mid(ct.Name, 4)
      ct.SetFocus
      ct.Text = ""
    End If
  Next
End Sub
```

```
Sub RemplitBDi()
  Dim NomTB As String, ct As Control
  For Each ct In Controls
    If Left(ct.Name, 3) = "TB_" Then
      NomTB = Mid(ct.Name, 4)
      ct.SetFocus
      ct.Text = Nz(DonMemb(NomTB), "")
    End If
  Next
End Sub

Function VerNomPren() As Boolean
  If (Nm = "") Or (Pr = "") Then
    MsgBox "Il faut un nom et un prénom"
    VerNomPren = False
  Else
    VerNomPren = True
  End If
End Function

Private Sub B_Annul_Click()
  Dernier = False
  Satisf = False
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub B_Correct_Click()
  B_Suppr.Enabled = True
  B_OK.Enabled = True
  B_OKDern.Enabled = True
End Sub

Private Sub B_OK_Click()
  If Not VerNomPren Then Exit Sub
  CaptureDon
  Dernier = False
  Satisf = True
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub B_OKDern_Click()
  If Not VerNomPren Then Exit Sub
  CaptureDon
  Dernier = True
  Satisf = True
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub

Private Sub B_Quit_Click()
  Dernier = True
  Satisf = False
  DoCmd.Close acForm, Me.Name, acSaveNo
End Sub
```

```
Private Sub B_Suppr_Click()
  Dim Rep
  Rep = MsgBox("Etes-vous sûr de vouloir supprimer ce membre ? ", _
    vbYesNo + vbQuestion)
  If Rep = vbYes Then
    Rst.Delete adAffectCurrent
    Rst.Update
    B_Annul_Click
  End If
End Sub

Function TestSQL(chSQL As String) As Boolean
  Dim Cm As New ADODB.Command, Res
  Set Cm.ActiveConnection = CurrentProject.Connection
  Cm.CommandType = adCmdText
  Cm.CommandText = chSQL
  Set Res = Cm.Execute
  If Res(0).Value <> 0 Then TestSQL = True Else TestSQL = False
  Set Cm = Nothing
End Function

Private Sub B_Ver_Click()
  Dim Ch As String, Rep
  If Mode = 0 Then
    Ch = "select count(Nom) from Membres " + _
        "where (Nom='" + Nm + "') and (Prénom='" + Pr + "')"
    If TestSQL(Ch) Then
      Rep = MsgBox("Ce nom et prénom sont déjà présents" + vbCr + _
        "voulez-vous tout de même entrer ce membre", vbYesNo + _
        vbExclamation)
      If Rep = vbYes Then B_OK.Enabled = True: B_OKDern.Enabled = True
    Else
      B_OK.Enabled = True: B_OKDern.Enabled = True
    End If
  Else                        'Mode
    If Not EnCours Then
      If Nm = "" Then
        MsgBox "Il faut un nom."
        Exit Sub
      End If
      Ch = "select count(Nom) from Membres where "
      If Pr = "" Then
        Ch = Ch + "InStr(Nom, '" + Nm + "')>0"
      Else
        Ch = Ch + "(Nom='" + Nm + "') and (Prénom='" + Pr + "')"
      End If
      If Not TestSQL(Ch) Then
        MsgBox "Pas d'enregistrement conforme. Changez ou annulez."
        Exit Sub
      End If
      EnCours = True
      Ch = Replace(Ch, "count(Nom)", "*")
      Rst.Open Ch, Cnx, adOpenDynamic, adLockOptimistic
      Rst.MoveFirst
    Else                 'Not EnCours
      Rst.MoveNext
```

```
      If Rst.EOF Then
        MsgBox "Plus d'enregistrements."
        EnCours = False
        VideBDi
        Rst.Close
        Exit Sub
      End If
    End If                'Not EnCours
    RecupChamps
    RemplitBDi
  End If                  'Mode
End Sub

Private Sub Form_Current()
  Nm = "": Pr = ""
  If Mode = 0 Then
    Me.Caption = "Nouveau membre"
    B_Ver.Caption = "Vérifier"
    B_Correct.Enabled = False
    B_Correct.Visible = False
    B_Suppr.Enabled = False
    B_Suppr.Visible = False
  Else
    Me.Caption = "Modification/Suppression membre"
    B_Ver.Caption = "Chercher/Suivant"
    B_Correct.Enabled = True
    B_Correct.Visible = True
    B_Suppr.Enabled = False
    B_Suppr.Visible = True
    EnCours = False
  End If
End Sub

Private Sub TB_Nom_Exit(Cancel As Integer)
  Nm = Nz(TB_Nom.Text, "")
  B_OK.Enabled = False
  B_OKDern.Enabled = False
End Sub

Private Sub TB_Prénom_Exit(Cancel As Integer)
  Pr = Nz(TB_Prénom.Text, "")
  B_OK.Enabled = False
  B_OKDern.Enabled = False
End Sub
```

3. PROCÉDURE MODIFMEMBRE ET ANNEXES

```
'-----------------------------------------------------RecupChamps--
Sub RecupChamps()
  Dim NomC As String, ff As Object
  DonMemb.RemoveAll
  For Each ff In Rst.Fields
    NomC = ff.Name
    If (NomC <> "N°") And (NomC <> "Cotis à jour") _
            Then DonMemb.Add NomC, ff.Value
    Next
End Sub
```

```
'-----------------------------------------------------RemplitChamps--
Sub RemplitChamps()
  Dim NomC As String, ff As Object
  For Each ff In Rst.Fields
    NomC = ff.Name
    If (NomC <> "N°") And (NomC <> "Cotis à jour") _
            Then ff.Value = DonMemb(NomC)
    Next
End Sub

'-----------------------------------------------------ModifMembre--
Sub ModifMembre()
  Dernier = False
  While Not Dernier
    Set DonMemb = CreateObject("Scripting.Dictionary")
    Set Cnx = CurrentProject.Connection
    Set Rst = New ADODB.Recordset
    Satisf = False
    Mode = 1
    DoCmd.OpenForm "F_Membre", , , , acFormEdit, acDialog
    If Satisf Then
      RemplitChamps
      Rst.Update
    End If
    DonMemb.RemoveAll: Set DonMemb = Nothing
    If Rst.State <> 0 Then Rst.Close
    Cnx.Close
    Set Rst = Nothing: Set Cnx = Nothing
  Wend
End Sub
```

La structure est semblable à `NouvMemb` mais certaines instructions comme l'ouverture du Recordset ne sont plus là : elles sont dans le module de classe car la recherche de l'enregistrement à modifier ou supprimer est en plusieurs épisodes séparés par clic sur Suivant. La différence importante est la valeur donnée à la variable `Mode` : une erreur ou l'oubli de cette instruction empêcherait le fonctionnement correct. Notez aussi que la fermeture du Recordset est conditionnelle : en effet, il peut être fermé par B_Ver_Click et si on essaie de le fermer une 2e fois, on se « paie » un message d'erreur.

Nous sommes parvenus au fichier *AmisAnimaux_3.accdb*. Pour essayer quelques modifications, vous cliquez sur le bouton et modifiez quelques données. Ensuite, vous fermez le formulaire Menu et ouvrez la table Membres pour vérifier que les modifications sont bien entrées et sont à la bonne place.

POUR ALLER PLUS LOIN

Voici quelques directions de possibles améliorations :

Offrir un système d'Aide

Offrir un système d'aide. Il faut bien sûr créer les fichiers .htm voulus ; ce n'est pas le sujet de ce livre. Ensuite, il faut fournir au moins un bouton dans le formulaire *Menu* un bouton dans le formulaire *F_Membre*. Nous avons vu comment écrire les routines de clic de ces boutons.

Gérer des rubriques supplémentaires

C'est simple en s'inspirant des routines écrites pour les rubriques en place.

Ajouter des fonctionnalités

Par exemple faire un système de relance des adhérents en retard de cotisation ; il faudrait ajouter la rubrique date de dernière cotisation… C'est utile pour toutes les associations.

Améliorer l'ergonomie

Si on clique sur Chercher/Suivant une fois de trop, le nom ne sera pas trouvé et il faudra reprendre la recherche au début. Il serait plus ergonomique d'implanter un bouton Précédent permettant des allers et retours.

Maintenant, quelques leçons à retenir de cette étude de cas (et de toutes) :

– Complémentarité de tous les éléments d'un projet : les instructions sont écrites en fonction de la structure des données dans les tables ; les procédures événements liés aux contrôles de BDi et les appels des BDi sont écrits en fonction les uns des autres et les transmissions de données doivent être prévues… Autre comportement de la BDi, autre façon de l'utiliser.
– Par ailleurs, le fait qu'on puisse ouvrir les tables directement sous Access offre un moyen indépendant de notre programme de les examiner et donc de vérifier ce que fait notre programme. Bien sûr, cet accès est surtout important pendant la phase de mise au point.